УДК 004.657

ОЦЕНКА ВРЕМЕНИ СОЕДИНЕНИЯ ДВУХ ТАБЛИЦ В ПАРАЛЛЕЛЬНОЙ КОЛОНОЧНОЙ СИСТЕМЕ БАЗ ДАННЫХ

Ю.А. Григорьев, **Е.Ю. Ермаков**

МГТУ им. Н.Э. Баумана, Москва
e-mail: grigorev@iu5.bmstu.ru; JK.Ermakov@gmail.com

Проанализированы существующие методы выполнения соединения отношений в параллельной колоночной системе баз данных. Выведено преобразование Лапласа–Стилтьеса времени соединения двух таблиц методом вложенных циклов, а также рассмотрены варианты этого преобразования для архитектур SE, SD, SN и различных режимов работы параллельных систем баз данных.

Ключевые слова: параллельные колоночные системы баз данных, преобразование Лапласа–Стилтьеса, математическое ожидание времени выполнения соединения отношений.

ESTIMATION OF TIME OF JOINING TWO TABLES IN THE PARALLEL COLUMN-ORIENTED DATABASE SYSTEM

Yu.A. Grigoriev, **Ye.Yu. Yermakov**

Bauman Moscow State Technical University, Moscow
e-mail: grigorev@iu5.bmstu.ru; JK.Ermakov@gmail.com

The existing methods for performing a join of relations in the parallel column-oriented database system are analyzed. The Laplace–Stieltjes transform is deduced for the time of joining two tables using the nested-loop method, and the variants of this transform are considered for the SE, SD, and SN architectures and different operating modes of the parallel database systems.

Keywords: parallel column-oriented database systems, Laplace–Stieltjes transform, expectation of time of performing a join of relations.

Российский бизнес все острее осознает необходимость построения хранилищ данных. Являясь одними из наиболее значимых элементов ИТ-инфраструктуры предприятия, хранилища консолидируют информацию, необходимую для создания достоверных аналитических отчетов. Они являются одними из крупнейших источников информации для современных аналитиков. По оценке Gartner, хранилища данных в ближайшей перспективе останутся одними из ключевых компонентов автоматизированных информационных систем предприятий [1]. Большой потенциал колоночных систем в области построения хранилищ данных подтверждают как аналитические исследования и прогнозы аналитиков [1–3], которые считают колоночные СУБД одним из основных и перспективных направлений развития, так и практическое использование таких систем для построения крупных хранилищ данных [4].

Перед архитектором информационной системы обработки данных возникает непростая задача выбора между традиционными (строчными — Oracle, MS SQL Server, MySql и др.) и специализированными

СУБД (колоночными — Vertica, ParAccel, MonetDB и др.). Для принятия обоснованного технического решения по выбору типа СУБД необходимо использовать средства моделирования. Для традиционных реляционных СУБД такие методы уже существуют [5]. Для параллельных СУБД подобные исследования ведутся [6–9], но находятся на начальной стадии развития.

Ранее в статье [10] были предложены математические методы оценки времени выполнения запроса к одной таблице в параллельной колоночной системе баз данных (ПКСБД). В настоящей работе предлагаются математические методы оценки времени выполнения соединения таблиц в ПКСБД, учитывающие особенности выполнения запросов к БД проектируемой системы, а также особенности реализации колоночных хранилищ. Получение такой оценки важно, так как операция соединения (join) часто используется в аналитических запросах к хранилищам данных, построенным на основе реляционных БД (ROLAP).

Организация работы колоночного хранилища. Под строчным хранением данных обычно понимается физическое хранение кортежа любого отношения в виде одной записи, в которой значения атрибутов идут последовательно одно за другим, а за последним атрибутом записи в общем случае следует новая запись отношения [11]. Таким образом, на физическом носителе отношение R представлено в следующем виде:

$$[\dot{a}_{11}, \dot{a}_{21}, \ldots, \dot{a}_{n1}]_1[\dot{a}_{12}, \dot{a}_{22}, \ldots, \dot{a}_{n2}]_2[\dot{a}_{13}, \dot{a}_{23}, \ldots, \dot{a}_{n3}]_3 \ldots \\ \ldots [\dot{a}_{1m}, a_{2m}, \ldots, \dot{a}_{nm}],$$

где $\dot{a}_{ij}$ — значение атрибута a_i в j-м кортеже отношения R; $[\dot{a}_{1j}, \dot{a}_{2j}, \ldots \ldots, \dot{a}_{nj}]_j$ — j-й кортеж отношения R; n — число атрибутов отношения R; $m = T(R)$ — число кортежей отношения R.

В колоночных хранилищах значения одного атрибута хранятся последовательно друг за другом [11], т.е. на физическом носителе отношение R примет следующий вид:

$$\langle \dot{a}_{11}, \dot{a}_{12}, \dot{a}_{13}, \ldots, \dot{a}_{1m}\rangle_1\langle \dot{a}_{21}, \dot{a}_{22}, \dot{a}_{23}, \ldots, \dot{a}_{2m}\rangle_2 \ldots \\ \ldots \langle \dot{a}_{n1}, \dot{a}_{n2}, \dot{a}_{n3}, \ldots, \dot{a}_{nm}\rangle_n,$$

где $\dot{a}_{ij}$ — значение атрибута a_i в j-м кортеже отношения R; $\langle \dot{a}_{i1}, \dot{a}_{i2}, \dot{a}_{i3}, \ldots, \dot{a}_{im}\rangle_i$ — i-й столбец (атрибут) отношения R.

Каждая колонка, хранимая на диске, разделена на блоки определенного размера. Блок состоит из заголовка, размер которого пренебрежимо мал по сравнению с размером блока и непосредственно данных. При одном запросе к диску происходит чтение нескольких блоков, число которых определяется некоторым параметром. Каждой записи в столбце сопоставляется ее позиция (номер строки). В большинстве

современных колоночных БД [12] значения столбца упорядочиваются по их позициям.

На логическом уровне колоночное хранилище выглядит идентично строчному: оно, по существу, представляет собой только модификацию физической (дисковой) структуры БД. В общем случае колоночные БД могут реализовывать совместимый со стандартами интерфейс реляционной БД (например, ODBC, JDBC и т.д.). Основные на данном уровне различия заключаются в информационных процессах, протекающих при формировании плана запроса и в процессе его выполнения [12]. Использование колоночных СУБД позволяет существенно уменьшить время реализации запросов к БД [10].

Материализация кортежей. Одним из процессов при формировании ответа на запрос в колоночных БД является материализация кортежей — процесс воссоздания кортежа на основе столбцов-атрибутов. В зависимости от момента применения данной операции в плане запроса существуют следующие варианты материализации [13]:

- *ранняя материализация*. Данный вариант аналогичен "естественной" материализации, применяемой в строчных хранилищах: каждый раз, когда осуществляется доступ к новому атрибуту, он добавляется к кортежу;
- *поздняя материализация*. Специфика колоночных хранилищ позволяет отложить процесс материализации до определенного момента, используя в процессе выполнения запроса позиции значений в колонках вместо самих значений атрибутов. К преимуществам данного метода можно отнести более высокую скорость работы с позициями значений по сравнению со всем кортежем.

Сжатие данных. В современных СУБД широко используется сжатие данных. Это позволяет повысить производительность уменьшая число дисковых операций ввода-вывода и объем передаваемых по сети данных. Колоночное хранение отношений позволяет улучшить этот показатель по сравнению со строчными хранилищами. Это достигается путем использования коэффициентов повторяемости значений атрибутов и возможности оперировать сжатыми данными (т.е. отсутствие затрат на декомпрессию). В работе [14] предлагается полученный эмпирическим путем алгоритм выбора типа компрессии данных в столбцах.

Параллельная обработка запросов. Основной формой параллельной обработки запросов является фрагментный параллелизм. Подробно данный процесс рассмотрен в работах [6–9, 15]. В соответствие с этой схемой запрос на языке SQL преобразуется в некоторый последовательный план. Данный последовательный план преобразуется в параллельный план, представляющий собой совокупность n идентичных параллельных агентов, которые реализуют те же операции, что

и последовательный план. Здесь n обозначает число процессорных узлов. Это достигается путем вставки оператора обмена exchange в соответствующие места дерева плана запроса. На завершающем этапе агенты рассылаются на соответствующие процессорные узлы, где параллельно интерпретируются исполнителями запросов. Результаты выполнения агентов объединяются корневым оператором exchange на нулевом процессорном модуле.

Рассмотрим процесс параллельной обработки запроса, где выполняется соединение таблиц R и S БД (рис. 1). $Q = R \rhd\lhd S$ — это логическая операция соединения (join) двух отношений (таблиц) R и S по некоторому общему атрибуту Y. В данном примере таблица R фрагментирована произвольным образом, а таблица S — по атрибуту соединения Y. На рис. 1 показано, что логический план выполнения соединения двух отношений тиражируется на n процессоров в параллельной системе БД (на рис. 1 показаны два процессора). Далее происходит параллельная обработка на каждом процессоре соответствующих фрагментов таблиц R и S. Вследствие того что таблица R не фрагментирована по атрибуту соединения, при последовательном чтении записей этой таблицы происходит их обработка в операторе

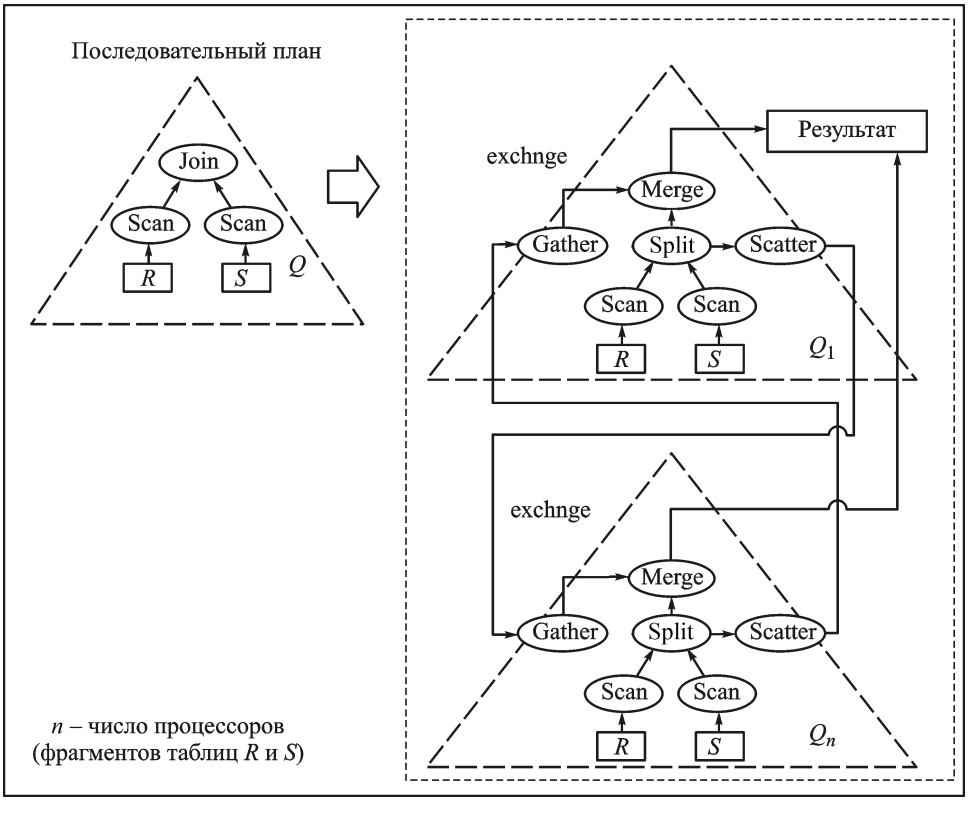

Рис. 1. Обработка запроса $Q = R \rhd\lhd S$ в параллельной системе БД

exchange, осуществляющим разбор записи и ее межпроцессорный обмен. Таблица S фрагментирована по атрибуту соединения, и записи, читаемые из фрагментов этой таблицы, обрабатываются на каждом процессоре локально.

Оператор exchange является составным оператором и включает в себя четыре оператора: Split, Gather, Scatter и Merge. Оператор Split — это оператор, который осуществляет разбиение кортежей (записей), поступающих из входного потока, на две группы: свои и чужие. Свои кортежи — это кортежи, которые должны быть обработаны на данном процессорном узле. Эти кортежи направляются в выходной буфер оператора Split (стрелка вверх). Чужие кортежи, т.е. кортежи, которые должны быть обработаны на процессорных узлах, отличных от данного, помещаются оператором Split во входной буфер правого дочернего узла, в качестве которого фигурирует оператор Scatter. Нульарный оператор Scatter извлекает кортежи из своего входного буфера и пересылает их на соответствующие процессорные узлы. Нульарный оператор Gather выполняет перманентное чтение кортежей из указанного порта со всех процессорных узлов, отличных от данного. Оператор Merge, реализующий логическую операцию join, определяется как бинарный оператор, который забирает кортежи из выходных буферов своих дочерних узлов, соединяет их и помещает результат в собственный выходной буфер. Таким образом, с помощью рассмотренных операций оператор exchange реализует полноценный межпроцессорный обмен записями в параллельной системе БД при обработке запроса методом фрагментного параллелизма.

Архитектуры параллельных систем БД. Наиболее распространенной системой классификации параллельных систем БД является система, предложенная Майклом Стоунбрейкером (Michael Stonebraker) [15]:

1. SE (Shared-Everything) — архитектура с разделяемыми памятью и дисками;

2. SD (Shared-Disks) — архитектура с разделяемыми дисками;

3. SN (Shared-Nothing) — архитектура без совместного использования ресурсов.

Пересечение отношений в колоночных хранилищах. При пересечении отношение во внешнем цикле остается отсортированным по порядку, отношение во внутреннем цикле — нет (рис. 2, здесь в результирующей таблице указаны номера позиций в исходных колонках). Поскольку чтение по неотсортированным позициям и последующее пересечение может занять продолжительное время, в [12] предлагается для внутреннего отношения использовать раннюю материализацию (в оператор пересечения сразу подаются готовые кортежи).

В параллельных БД нефрагментированная по атрибуту соединения таблица пересылается другим узлам и используется во внешнем цикле. Поскольку оператор exchange работает покортежно [15], передавать битовые маски нельзя и для обоих отношений необходимо использовать раннюю материализацию.

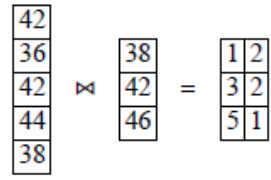

Рис. 2. Иллюстрация пересечения

Если оба отношения фрагментированы по атрибуту соединения, то для внешнего отношения возможны поздняя материализация и обработка с помощью битовых масок.

Преобразование Лапласа–Стилтьеса времени соединения двух таблиц. Ниже получены преобразования Лапласа–Стилтьеса (ПЛС) времени соединения таблиц A и B в параллельной системе баз данных в соответствии с планом $\pi_A(\sigma_{F_A}(A)) \triangleright\triangleleft \pi_B(\sigma_{F_B}(B))$ и условиями поиска (фильтрации) в исходных таблицах $F_A = f_{A0} \cap \bigcap_{i=1}^{|K_{AF}|} f_{Ai}$, $F_B = f_{B0} \cap \bigcap_{i=1}^{|K_{BF}|} f_{Bi}$, где π — операция проекции, σ — операция селекции, $\bowtie$ — операция естественного соединения подзапросов; f_{A_i} (f_{B_i}) — элементарное условие поиска по i-му атрибуту таблицы A (B), например $a_i > 5$; K_{AF} (K_{BF}) — множество таких атрибутов таблицы A (B), на которые накладываются элементарные условия поиска; f_{A_0} (f_{B_0}) — условие поиска, которое включает в себя сравнение разных атрибутов таблицы A (B), например $a_i > a_j$.

Преобразования Лапласа–Стилтьеса позволяют оценивать не только математические ожидания случайных величин, но и моменты более высоких порядков, например дисперсию. Предполагается, что записи таблиц фильтруются и соединяются по неиндексированным атрибутам (т.е. рассматривается наихудший случай). Также предполагается, что таблица B фрагментирована по атрибуту соединения, а таблица A — нет. Преобразование Лапласа–Стилтьеса получено для метода NLJ (соединение с помощью вложенных циклов). Введем следующую функцию для i-го узла:

$$V_{Ai}(s, Z) = G_{A_i}(\chi_1(s, r_1, m_1) \cdot \Psi_{A_1}(s, Z)), \tag{1}$$

где

$$G_{A_i}(z) = z^{\frac{V_A}{n}} \tag{2}$$

— производящая функция числа записей исходной фрагментированной таблицы A (не по атрибуту соединения), обрабатываемых в i-м процессоре (узле); V_A/n — число записей таблицы A, которые хранятся в i-м узле; n — число узлов; функция $\Psi_{A_1}(s, Z)$ учитывает, что читаются кортежи колонок по позициям, которые удовлетворяют условиям

поиска по предыдущим атрибутам, она рекуррентно определяется следующим образом:

$$\Psi_{A_1}(s, \mathrm{Z}) = \Omega(P_{Af_1}, \chi_2(s, r_2, m_2) \cdot \Psi_{A_2}(s, Z)), \tag{3}$$

. . . . . .

$$\Psi_{A_i}(s, \mathrm{Z}) = \Omega(P_{Af_i}, \chi_{i+1}(s, r_{i+1}, m_{i+1}) \cdot \Psi_{A(i+1)}(s, \mathrm{Z})),$$

. . . . . .

$$\Psi_{A_b}(s, Z) = \Omega(P_{Af_b}, \phi_P^{u_A}(s) \cdot \Omega(P_{AT}, \Psi_{A\pi\sigma}(s) \cdot q_{Ain}(\mathrm{Z})));$$

$$Z = (z_1, \ldots, z_n); \tag{4}$$

$$\Omega(P, z) = 1 - P(1 - z)_; \tag{5}$$

$b = |K_{AF}|$ — число атрибутов отношения A, по которым происходит фильтрация кортежей по условию $\bigcap\limits_{i=1}^{|K_{A_F}|} f_{A_i}$;

$$\Psi_{A\pi\sigma}(s) = \prod_{j=|K_{A_F}|+1}^{|K_{A_F}|+|K_{A\pi}|+|K_{A\sigma}|} \chi_j(s, 1, m_j) \tag{6}$$

— ПЛС времени чтения значений из колонок по значениям позий (т.е. по смещению); $K_{A\pi}$ — подмножество атрибутов отношения A, которые участвуют в операции проекции и не присутствуют в множестве K_{A_F}; K_{A_σ} — подмножество атрибутов отношения A, которые участвуют в операции соединения и не присутствуют в множестве K_{A_π};

$$\chi_i(s, r, m) = \phi_{Di}(s)\phi_M^{m\nu_i}(s)\phi_P^r(s); \tag{7}$$

$\phi_{D_i}(s)$ — ПЛС времени чтения кортежа i-го столбца с диска; $\phi_M^{m\nu_i}(s)$ — ПЛС времени сохранения атрибута в оперативной памяти (ОП) и его чтения в кэш процессора; ν_i — размер атрибута; m — число циклов чтения/записи в ОП (на байт), необходимых для проверки условия по соответствующему атрибуту (для i-го столбца вводится аргумент m_i — см. (1), (3)); $\phi_P^r(s)$ — ПЛС времени обработки кортежа столбца в процессоре, r — число логических операций, необходимых для проверки условия по соответствующему атрибуту (для i-го столбца вводится аргумент r_i — см. (1), (3)); $\chi_j(s, 1, m_j)$ — ПЛС времени чтения кортежа j-го столбца проекции (или селекции) с диска в кэш процессора и обработки в нем (см. (6)); 1 означает, что в процессоре проверяется только значение битовой маски в соответствующей позиции; u_A — число логических операций процессора, необходимых для проверки условия f_{A_0}; P_{AT} — вероятность того, что сформированный кортеж удовлетворяет условию f_{A_0}; $q_{Ain}(z_1, \ldots, z_n)$ — производящая функция,

учитывающая межпроцессорный обмен, она определяется рекуррентными формулами

$$\begin{aligned}
&q_{Ain}(z_1,\ldots,z_n) = (1-p_{Ain})q_{Ain-1}(z_1,\ldots,z_{n-1}) + p_{Ain}z_n,\\
&q_{Ain-1}(z_1,\ldots,z_{n-1}) =\\
&\qquad = (1-p_{Ain-1})q_{Ain-2}(z_1,\ldots,z_{n-2}) + p_{Ain-1}z_{n-1},\\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\
&q_{Ai1}(z_1) = (1-p_{Ai1}) + p_{Ai1}z_1, p_{Ai1}\equiv 1
\end{aligned} \tag{8}$$

(p_{Aij} — это вероятность, что запись передается из i-го узла в j-й узел при условии, что она не была передана в узлы $n,\ldots,j+1$);

$$V_{A_i}(s,z_1,\ldots,z_n)\Big|_{s=0} = G_{A_i}(\Omega(P_{AC}\cdot P_{AT}, q_{Ain}(z_1,\ldots,z_n))) \tag{9}$$

— производящая функция (ПФ) числа записей исходной фрагментированной таблицы A i-го узла, передаваемых другим процессорам в соответствии с функцией распределения этого узла; $P_{AC} = \prod\limits_{i=1}^{|K_{AF}|} P_{Af_i}$ — вероятность того, что сформированный кортеж удовлетворяет элементарным условиям по атрибутам K_{AF} (условие $\bigcap\limits_{i=1}^{|K_{AF}|} f_{Ai}$);

$$W_{A_i}(z) = \prod_{j=1}^{n} V_{A_j}(0,\ 1_1,\ldots,1_{i-1},z,\ 1_{i+1},\ldots,1_n) \tag{10}$$

— ПФ числа записей таблицы A, соединяемых в i-м узле.

Производящая функция результирующего числа записей, полученных после соединения таблиц A и B в i-м узле, определяется следующим выражением:

$$W_{AB_i}(z) = W_{A_i}(G_{B_i}(\Omega(P_{BC}\cdot P_{BT}\cdot P_{AB},\ z))); \tag{11}$$

здесь $G_{B_i}(z) = z^{\frac{V_B}{n}}$ — ПФ числа записей исходной фрагментированной таблицы B, обрабатываемых в i-м узле; V_B/n — число записей таблицы B, которые хранятся в i-м узле. Предполагается, что таблица B фрагментирована по атрибуту соединения, P_{AB} — вероятность, что две записи из таблиц A и B удовлетворяют условию соединения, т.е.

$$P_{AB} = \sum_{j=1}^{|D_A|} \eta_{A_j}\eta_{B_k}; \tag{12}$$

здесь $|D_A|$ — мощность домена атрибута соединения в таблице A, η_{A_j} — вероятность, что атрибут соединения в записи таблицы A принимает значение $d_{A_j} \in D_A$, η_{B_k} — вероятность, что атрибут соединения в

записи таблицы B принимает значение $d_{B_k} \in D_B : d_{B_k} = d_{A_j}$, D_B — домен атрибута соединения в таблице B.

Получим формулу для ПЛС времени соединения таблиц по методу NLJ в i-м узле. Введем следующую функцию:

$$H_{A_i}(s,z) = \\ V_{A_i}(s,\, \phi_N^{w_A\cdot\nu_A}(s),\ldots,\phi_N^{w_A\cdot\nu_A}(s),\overset{i}{z},\phi_N^{w_A\cdot\nu_A}(s),\ldots,\phi_N^{w_A\cdot\nu_A}(s))\times \\ \times\prod_{j=1,j\neq i}^{n} V_{Aj}(0,\,1_1,\ldots,1_{i-1},z,1_{i+1},\ldots,1_n), \tag{13}$$

где $V_{A_i}(\cdot)$ определяется выражением (1).

Функция (13) определяет ПЛС времени чтения записей таблицы A в i-м узле (s), ПЛС времени пересылки записей таблицы другим узлам по команде exchange ($\phi_N^{w_A\cdot\nu_A}(s)$), где они обрабатываются процессорами принимающих узлов, а также ПФ числа записей таблицы A, соединяемых в i-м узле (z); ν_A — размер сформированного кортежа $\Big(\sum\limits_{i=1}^{|K_{A\pi}|}\nu_i\Big)$, w_A — число операций чтения/записи (на байт), которое необходимо для перемещения сформированных записей.

Преобразование Лапласа–Стилтьеса времени соединения таблиц по методу NLJ имеет вид

$$\phi_i^{NLJ}(s) = G_{Bi}(\chi_1(s,r_1,m_1)\cdot\Psi_{B1}(s,1))\times \\ \times H_{A_i}(s,G_{B_i}(\Psi_{B_1}(0,\phi_P^{u_{NLJ}}(s)\cdot\Omega(P_{AB},\phi_M^{w_{AB}\cdot\nu_{AB}}(s)\cdot\phi_N^{w_{AB}\cdot\nu_{AB}}(s))))), \tag{14}$$

функция $\Psi_{B_1}(s)$ рекуррентно определяется следующим образом:

$$\begin{aligned}
&\Psi_{B_1}(s,z) = \Omega(P_{Bf1},\chi_2(s,r_2,m_2)\cdot\Psi_{B2}(s,z)),\\
&\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\\
&\Psi_{Bi}(s,z) = \Omega(P_{Bfi},\chi_{i+1}(s,r_{i+1},m_{i+1})\cdot\Psi_{B(i+1)}(s,z)),\\
&\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\\
&\Psi_{B_c}(s,z) = \Omega(P_{Bfc},\phi_P^{u_B}(s)\cdot\Omega(P_{BT},\Psi_{B\pi\sigma}(s)\cdot z)),
\end{aligned} \tag{15}$$

$c = |K_{BF}|$ — число атрибутов отношения B, по которым происходит фильтрация кортежей по условию $\bigcap\limits_{i=1}^{|K_{BF}|} f_{Bi}$; ν_{AB} — размер сформированного кортежа $\Big(\sum\limits_{i=1}^{|K_{A\pi}|}\nu_i + \sum\limits_{i=1}^{|K_{B\pi}|}\nu_i\Big)$; w_{AB} — число операций чтения/записи (на байт), которое необходимо для перемещения сформированных записей.

Первый сомножитель в (14) определяет ПЛС времени формирования материализованного представления таблицы B, которое соединяется с записями таблицы A в i-м узле.

Анализ режимов работы системы.

1. Пакетный режим — режим offline (система рассматривается как замкнутая).

Предполагается, что "узкое место" — это диск. Из работы [10, формула в табл. 2] следует, что ПЛС времени чтения кортежа столбца с диска равно

$$\phi_D(s) = \Big(1 - \frac{1}{L}\Big) + \frac{1}{L}\Big((1 - p_D) + p_D\,\eta_{DB}(s)\Big), \tag{16}$$

где L — число позиций в блоке столбца, $1 - p_D$ — вероятность, что блок находится в буфере.

Далее речь пойдет об определении ПЛС времени чтения блока с диска с учетом очереди к диску, т.е. $\eta_{DB}(s)$.

В ПКСБД обрабатываются пакеты запросов. Их число — случайная величина $\xi > 0$, $\Theta(z)$ — ПФ ξ. В каждом пакете SQL-запросы выполняются последовательно (предполагается, что они связаны по данным: выходные данные одного запроса являются входными данными другого). Но запросы разных пакетов (по одному из каждого пакета) могут обрабатываться параллельно. Более того, каждый из этих запросов обрабатывается параллельно на n процессорах.

Рассмотрим наихудший случай, когда все n процессоров при обработке ξ запросов обращаются к дисковой системе, состоящей из N_D дисков (обработка как бы "проваливается" в дисковую систему).

Некоторый i-й процессор обращается к конкретному диску для чтения блока, связанного с обработкой запроса j-го пакета. Производящая функция числа других запросов, для которых читаются блоки с того же диска, равна

$$\text{ПФ}(z) = \frac{\Theta^n\Big(\Omega\Big(\frac{1}{N_D}, z\Big)\Big)}{\Omega\Big(\frac{1}{N_D}, z\Big)}. \tag{17}$$

Действительно, $\dfrac{\Theta^n(z)}{z}$ — это ПФ числа обрабатываемых запросов без 1 (т.е. без данного ij-го запроса); $\Omega\Big(\dfrac{1}{N_D}, z\Big) = 1 - \dfrac{1}{N_D}(1 - z)$ — учитывает, что для произвольного запроса его блок считывается с данного диска с вероятностью $1/N_D$ (испытание Бернулли).

Тогда ПЛС времени чтения блока (чередования) для ij-го запроса с учетом очереди к диску будет равна

$$\eta_{DB}(s) = \frac{\Theta^n\Big(\Omega\Big(\frac{1}{N_D}, \varphi_{DB}(s)\Big)\Big)}{\Omega\Big(\frac{1}{N_D}, \varphi_{DB}(s)\Big)} \cdot \varphi_{DB}(s), \tag{18}$$

где

$$\varphi_{DB}(s) = \frac{\mu_{DB}}{\mu_{DB} + s}, \tag{19}$$

$1/\mu_{DB}$ — среднее время чтения блока с диска.

Формула (18) справедлива для всех процессоров в очереди к диску, так как при чтении с диска используется "лифтовый" поиск, когда блоки читаются за один проход гребенки головок. На один поиск тратится следующее время: время подвода головок, время половины оборота шпинделя, время чтения блока.

2. Режим "запрос–ответ" — режим online (система рассматривается как разомкнутая). Положим, что i-я рабочая станция обращается к j-му запросу с интенсивностью λ_{ij}. Если предположить, что эти входные потоки заявок являются пуассоновскими, время обслуживания в ресурсах распределено по экспоненциальному закону, а переход от ресурса к ресурсу выполняется по вероятности, то модель обработки запросов можно представить в виде сети массового обслуживания (СМО). В этой сети обработку в узлах ресурсов можно представить в виде совокупности независимых СМО M/M/1 (это доказывается в теории массового обслуживания в виде теоремы разложения Джексона).

Ранее было получено ПЛС времени выполнения запросов (на одном из n процессоров после фрагментации таблиц — см. формулу (14)), которое зависит от ПЛС времени обработки в ресурсах системы:

$$\phi(n,\ \eta_{DB}(s), \phi_M(s), \phi_N(s), \phi_P(s)). \tag{20}$$

Здесь и далее будем опускать верхний (NLJ) и нижний (i) индексы в левой части (14); ПЛС, входящие в (20), должны быть определены в соответствии с разложением Джексона следующими выражениями:

$$\eta_{DB}(s) = \frac{\mu_{DB} - \lambda_{DB}}{\mu_{DB} - \lambda_{DB} + s} \text{ (см. также (16))},\ \phi_M(s) = \frac{\mu_M - \lambda_M}{\mu_M - \lambda_M + s},$$

$$\phi_N(s) = \frac{\mu_N - \lambda_N}{\mu_N - \lambda_N + s}, \quad \phi_P(s) = \frac{\mu_P - \lambda_P}{\mu_P - \lambda_P + s}. \tag{21}$$

Вопрос только в том, как определить интенсивности λ в формулах (21). Каждую интенсивность можно вычислить по формуле

$$\lambda_X = \sum_{i,j} \lambda_{ij} q_{Xj}, \tag{22}$$

где λ_{ij} — интенсивность обращения i-й рабочей станции к j-му запросу, q_{X_j} — среднее число обрабатываемых кортежей (или циклов, операций) в ресурсе X (DB — диск, M — ОП, N — сеть, P — процессор) при выполнении одного j-го запроса.

Для определения q_{X_j} ($X = DB, M, N, P$) можно использовать следующий прием. Заменим переменную s в выражениях (21) соответ-

ственно на переменные s_{DB}, s_M, s_N, s_P. Далее перепишем формулу (20) в следующем виде:

$$\phi(n,\, s_{DB}, \mu_{DB}, \lambda_{DB},\, s_M, \mu_M, \lambda_M,\, s_N, \mu_N, \lambda_N, s_P, \mu_P, \lambda_P). \qquad (23)$$

Для ресурса X величину q_{X_j}, которая используется в формуле (22), можно рассчитать, используя следующий алгоритм:

1. Положить в (23) $n = 1$, $\mu_X = 1$, $\lambda_X = 0$.

2. Для всех остальных ресурсов, кроме X, положить в (23) $s = 0$, $\lambda = 0$ и $\mu = 1$.

3. Найти частную производную $Q = \left.\dfrac{\partial \phi}{\partial s_X}\right|_{s_X=0}$.

4. Положить $q_{X_j} = \dfrac{Q}{n_X}$, где n_X — общее число узлов ресурса X во всей системе (для $X = DB$ — это общее число дисков, которые могут параллельно обрабатывать заявки, для $X = M$ — это общее число блоков памяти, которые могут параллельно обрабатывать заявки, для $X = N$ — это общее число параллельно работающих каналов межпроцессорной шины, для $X = P$ — это общее число процессоров в системе).

Оценка среднего времени выполнения запроса. Формулы для $\phi_{Di}(s)$ (i — номер колонки, т.е. атрибута), $\phi_M(s)$, $\phi_N(s)$, $\phi_P(s)$ в зависимости от архитектуры ПКСБД и режима работы представлены в табл. 1.

Дифференцируя выражение (14) как сложную функцию по s в нуле, можно получить моменты случайного времени (ξ) обработки запроса на соединение таблиц в ПКСБД:

$$M_\xi = -\phi'(0), \quad M_{\xi^2} = \phi''(0), \quad \sigma_\xi^2 = M_{\xi^2} - M_\xi^2. \qquad (24)$$

Здесь индексы в левой части (14) опущены. Ручное дифференцирование выражения (14) является весьма трудоемкой задачей. Для получения значений моментов можно использовать методы численного дифференцирования, описанные в [16]:

$$\phi'(x_0) \approx \frac{\left(\phi_{1/2}^1 - \dfrac{1}{2}\phi_1^2\right)}{h}, \quad \phi''(x_0) \approx \frac{\left(\phi_0^2 - \dfrac{1}{12}\phi_0^4\right)}{h^2}, \qquad (25)$$

где

$$\phi_i^m = \sum_{j=0}^{m} (-1)^j C_m^j \phi_{i+m/2-j} \qquad (26)$$

— разность m-го порядка (при четном m i — целое, при нечетном m i — полуцелое),

$$\phi_i = \phi(x_i) = \phi(x_0 + ih) \qquad (27)$$

— соответствующие значения функции, h — шаг таблицы разностей.

Таблица 1

Формулы для ПЛС времени пребывания в ресурсах: диске, ОП, сети, процессоре

Режим		$\phi_{Di}(s)$	$\phi_M(s)$	$\phi_N(s)$	$\phi_P(s)$
Online	SE	$\left(1-\frac{1}{L_i}\right)+\frac{1}{L_i}\left((1-p_D)+p_D\frac{\mu_{DB}-\lambda_{DB}}{\mu_{DB}-\lambda_{DB}+s}\right)$	$\frac{\mu_M-\lambda_M}{\mu_M-\lambda_M+s}$ (обмен между процессорами осуществляется через ОП)		$\frac{\mu_P-\lambda_P}{\mu_P-\lambda_P+s}$
	SD		$\frac{\mu_M-\lambda_M}{\mu_M-\lambda_M+s}$	$\frac{\mu_N-\lambda_N}{\mu_N-\lambda_N+s}$	
	SN				
Offline		$\left(1-\frac{1}{L_i}\right)+\frac{1}{L_i}\left((1-p_D)+p_D\,\eta_{DB}(s)\right)$ $\eta_{DB}(s)=\frac{\Theta^n\left(\Omega\left(\frac{1}{N_D},\varphi_{DB}(s)\right)\right)}{\Omega\left(\frac{1}{N_D},\varphi_{DB}(s)\right)}\cdot\varphi_{DB}(s)$ $\varphi_{DB}(s)=\frac{\mu_{DB}}{\mu_{DB}+s}$	$\frac{\mu_M}{\mu_M+s}$	$\frac{\mu_N}{\mu_N+s}$	$\frac{\mu_P}{\mu_P+s}$

Пример расчета. Далее приведен расчет времени выполнения соединения двух таблиц в ПКСБД. Характеристики ресурсов (интенсивности обработки) были получены с помощью программы синтетических тестов AIDA64 [17]. Расчеты выполнены при следующих значениях характеристик ресурсов.

1. Процессор – Intel Core i7-920 2.79 GHz. Для выбранного процессора измеренное значение числа процессорных циклов, выполняемых в секунду, $\mu_P = 2{,}79 \cdot 10^9$ 1/с.

2. Внешняя память $N_D = 250$, диск 3.5" Seagate Cheetan 15K.6 ST3146356FC; размер блока чередования (stripe size) $Q_{\text{БЧ}}$=64 кБ; среднее время поиска и чтения блока чередования с диска $t_{\text{БЧ}}=t_{\text{подвода}}+ + t_{\text{вращ}}/2 + Q_{\text{БЧ}}/v_{\text{чтения}} = 4 + 4/2 + 64/200 = 6{,}6$ мс. Поэтому интенсивность чтения блоков с диска равна $\mu_{DB} = 1000/6{,}3 = 160\,\text{с}^{-1}$, $p_D = 0{,}9$.

3. Оперативная память DDR3-1600 PC3 – 12800. Интенсивность чтения одного байта информации из ОП равна $\mu_M = 7500 \cdot 1024 \times \times 1024$ 1/с.

4. Параметры отношений и оператора соединения, использованных для расчетов, приведены в табл. 2. В скобках указаны операции, в которых участвует атрибут: f – при поиске кортежей, π – в операции проекции, σ – в операции селекции.

Ниже приведены графики зависимостей среднего времени соединения таблиц A и B от интенсивности поступления заявок на обра-

Таблица 2

Отношение A			Отношение B		
$V_A = 1000$	$P_{AT} = 0{,}01$		$V_B = 10000$	$P_{BT} = 0{,}01$	
$u_A = 700$	$w_A = 1$		$U_B = 700$	$W_B = 1$	
$A_{\text{тр.}}a_0$	(f)	$P = 0{,}33$	$A_{\text{тр.}}b_0$	(f)	$P = 0{,}33$
$A_{\text{тр.}}a_1$	(f)	$P = 0{,}33$	$A_{\text{тр.}}b_1$	(f)	$P = 0{,}33$
$A_{\text{тр.}}a_2$	(π)	$P = 1$	$A_{\text{тр.}}b_2$	(f)	$P = 1$
$A_{\text{тр.}}a_3$	$(f\sigma\pi)$	$P = 0{,}33$	$A_{\text{тр.}}b_3$	$(f\sigma\pi)$	$P = 0{,}33$
$A_{\text{тр.}}a_4$	(π)	$P = 1$	$A_{\text{тр.}}b_4$	(π)	$P = 1$
$A_{\text{тр.}}a_5$	(π)	$P = 1$	$A_{\text{тр.}}b_5$	(π)	$P = 1$
$A_{\text{тр.}}a_6$	$(-)$	$P = 1$	$A_{\text{тр.}}b_6$	(f)	$P = 0{,}33$
$A_{\text{тр.}}a_7$	$(-)$	$P = 1$	$A_{\text{тр.}}b_7$	$(-)$	$P = 1$
$A_{\text{тр.}}a_8$	$(-)$	$P = 1$	$A_{\text{тр.}}b_8$	$(-)$	$P = 1$
$A_{\text{тр.}}a_9$	$(-)$	$P = 1$	$A_{\text{тр.}}b_9$	$(-)$	$P = 1$

Примечание. Для всех атрибутов $\nu = 100$, $m = 1$, $r = 70$.

ботку соответствующего оператора Select, числа процессоров и других параметров.

1. Архитектура SE — режим online. На рис. 3, 4 представлены зависимости времени выполнения соединения от интенсивности заявок и числа процессоров в архитектуре SE. Из графиков следует, что при числе процессоров более 4 последующее их увеличение не приводит к существенному уменьшению времени обработки и к повышению предельного порога интенсивности заявок, после которого возникает перегрузка.

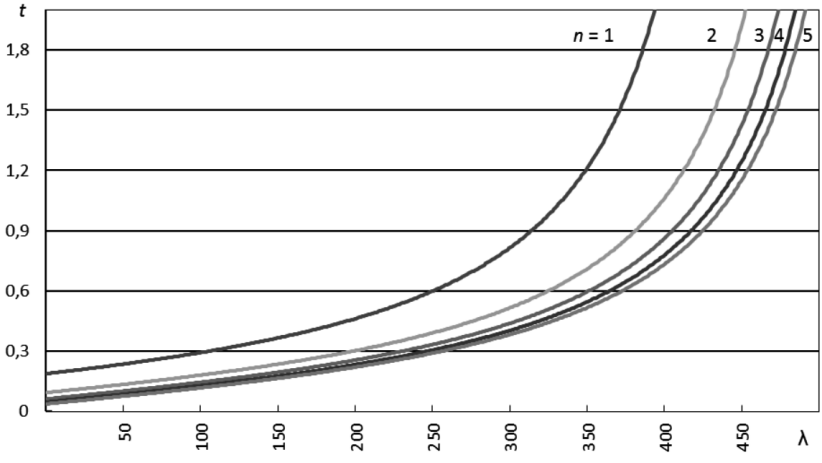

Рис. 3. Зависимость времени выполнения соединения от интенсивности запросов λ, параметр — число процессоров n

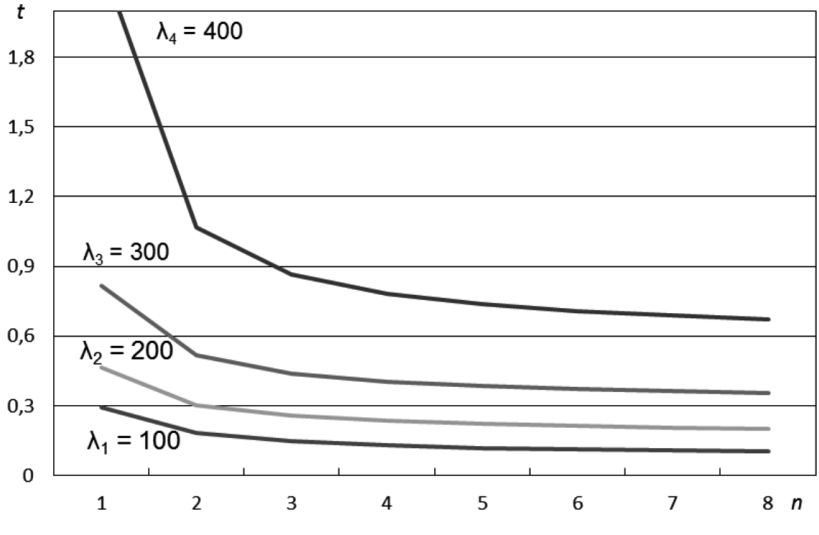

Рис. 4. Зависимость времени выполнения запроса от числа процессоров n, параметр — интенсивность заявок λ

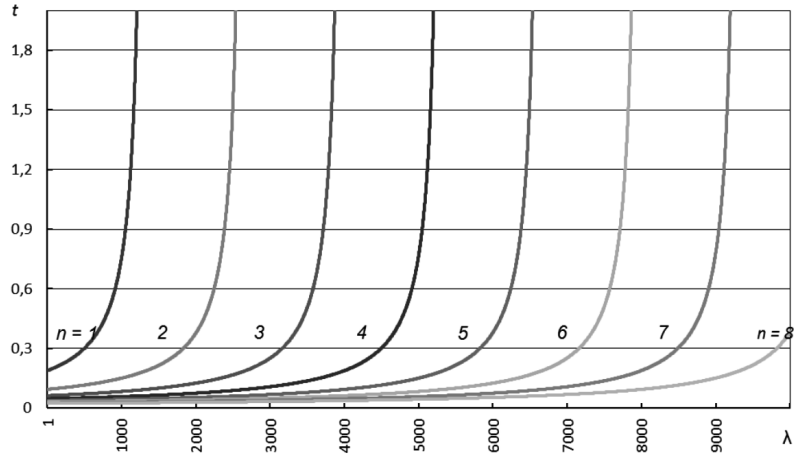

Рис. 5. Зависимость времени выполнения запроса от интенсивности запросов λ, параметр — число процессоров n

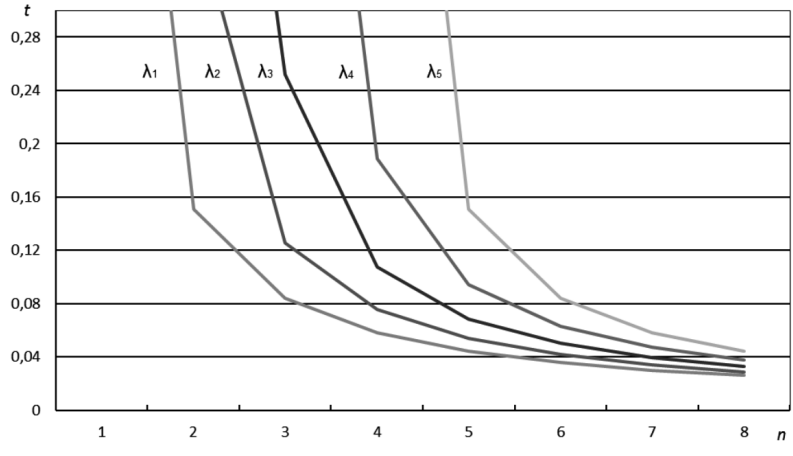

Рис. 6. Зависимость времени выполнения запроса от числа процессоров n, параметр — интенсивность запросов λ:
$\lambda_1 = 1000$, $\lambda_2 = 2000$, $\lambda_3 = 3000$, $\lambda_4 = 4000$, $\lambda_5 = 5000$

2. Архитектура SN — режим online. На рис. 5, 6 представлены зависимости среднего времени выполнения соединения таблиц от интенсивности заявок и числа процессоров в архитектуре SN. Так же, как и для режима SE, с ростом числа узлов темпы снижения времени выполнения запроса замедляются, но при этом возрастает предельный порог интенсивности. При этом предельные значения интенсивности почти в 5 раз превосходят аналогичные значения для архитектуры SE.

3. Пакетный режим. На рис. 7, 8 представлены зависимости времени выполнения соединения от числа пакетов и числа процессоров для пакетного режима работы. Из графиков следует, что время обработки практически линейно зависит от числа обрабатываемых пакетов ξ.

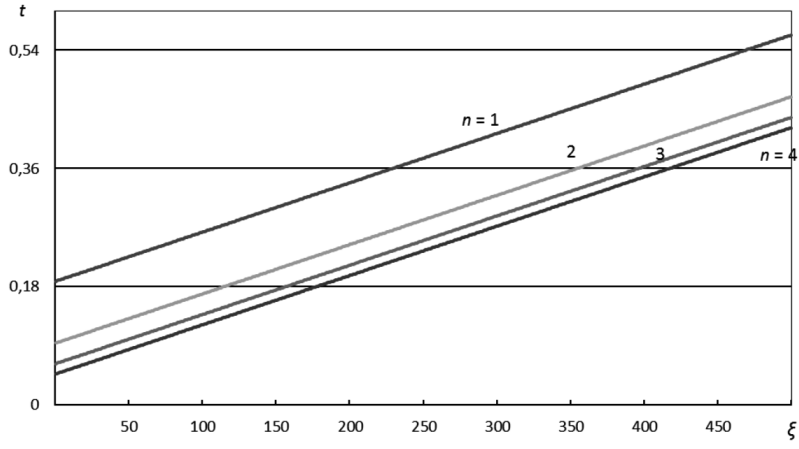

Рис. 7. Зависимость времени выполнения соединения от числа пакетов ξ, параметр — число процессоров n

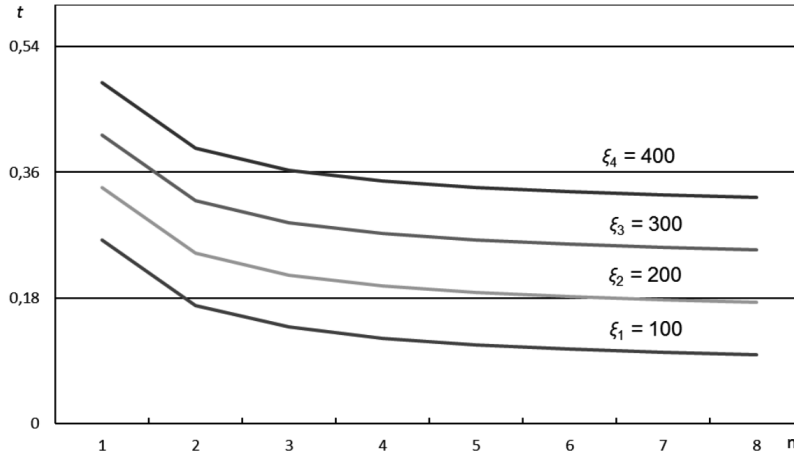

Рис. 8. Зависимость времени выполнения соединения от числа узлов n, параметр — число пакетов ξ

4. Сравнение режимов работы. На рис. 9, 10 представлены зависимости времени выполнения соединения таблиц от числа процессоров (узлов) для различных архитектур, режимов и нагрузок. Из графиков следует, что при единичной нагрузке ($\lambda = 1$ и $\xi = 1$) в пакетном режиме запросы обрабатываются дольше. При высоких нагрузках SN-архитектура показывает лучшее время по сравнению с архитектурами SD и SE.

На рис. 11 приведена зависимость производительности системы от числа процессоров (показатель масштабируемости) для различных архитектур и режимов работы. Видно, что при пакетном режиме производительность отклоняется от линейной, а в режиме online она близка

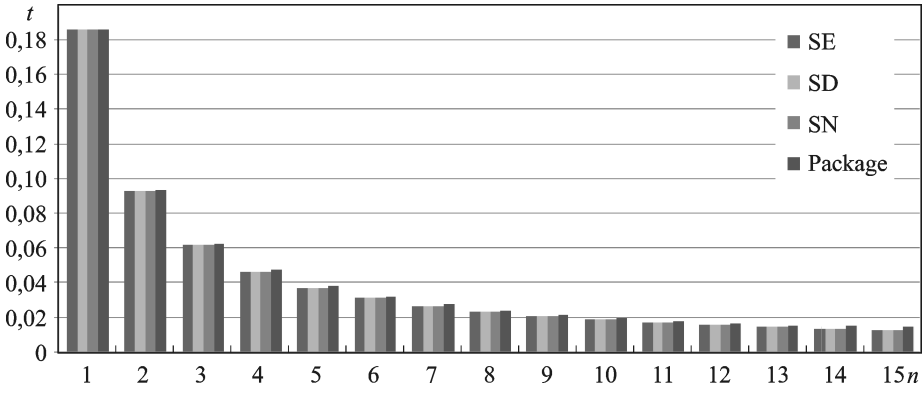

Рис. 9. Зависимость времени выполнения запроса от числа процессоров (узлов) для различных режимов работы при единичной нагрузке

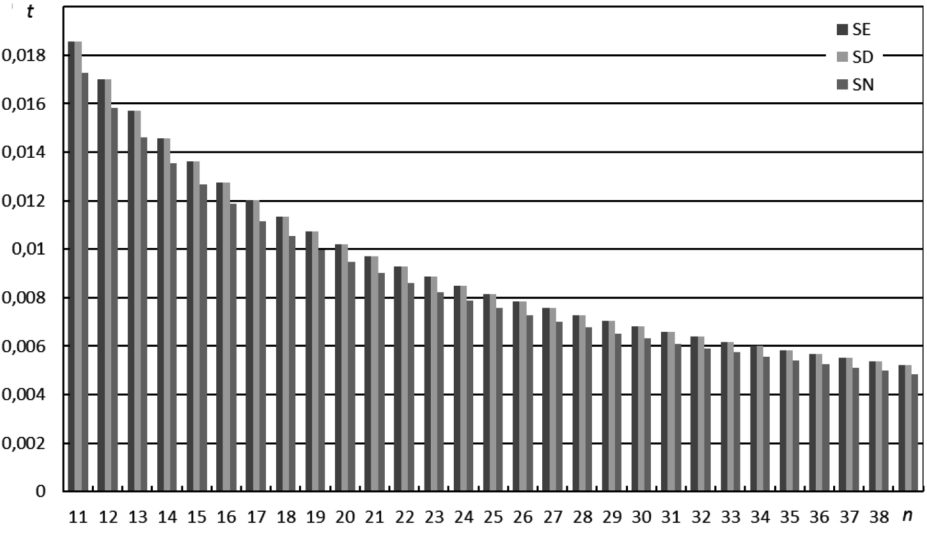

Рис. 10. Зависимость времени выполнения соединения от числа процессоров (узлов) для различных архитектур при нагрузке 100 заявок в секунду

к линейной (т.е. время выполнения запроса уменьшается практически пропорционально увеличению числа узлов). Но это, конечно справедливо, если с ростом числа процессоров накладные расходы ПКСБД увеличиваются незначительно.

Заключение. 1. Проанализирован процесс выполнения соединения двух таблиц в ПКСБД, реализованный на основе фрагментного параллелизма.

2. Получено ПЛС времени выполнения запроса на соединение двух таблиц в ПКСБД, позволяющее оценивать моменты различных порядков (математическое ожидание, дисперсию и др.).

3. Выполнен анализ пакетного режима функционирования ПКСБД и режима “запрос-ответ”.

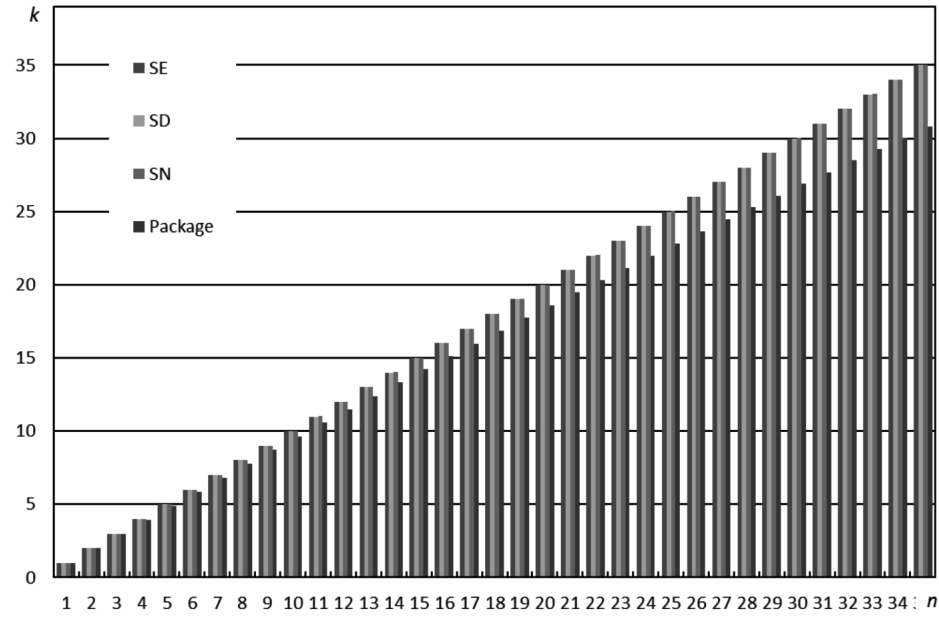

Рис. 11. Показатель масштабируемости системы при различных режимах работы

4. Приведен пример расчета и построены графики зависимостей среднего времени соединения таблиц от интенсивности поступления заявок на обработку соответствующего оператора Select, числа процессоров и других параметров для различных архитектур ПКСБД и режимов ее функционирования. Сделан ряд нетривиальных выводов.

5. В дальнейшем предполагается использовать аппарат ПЛС для оценки времени выполнения аналитических запросов к хранилищу данных, реализованному на основе ПКСБД и использующему специальные планы соединения таблиц измерений и фактов.

СПИСОК ЛИТЕРАТУРЫ

1. Арсентьев А. Хранилища данных становятся инфраструктурным компонентом № 1. CNews аналитика. 2010. [Электронный ресурс]. [http://retail.cnews.ru/reviews/free/BI2010/articles/articles6.shtml]. Проверено 27.06.2011.
2. Michael Stonebraker. My Top 10 Assertions About Data Warehouses / Перевод Сергея Кузнецова, 2010 г.: [Электронный ресурс]. [http://citforum.ru/gazeta/166/]. Проверено 27.06.2011.
3. Michael Stonebraker, Chuck Bear, Ugur Çetintemel, Mitch Cherniack, Tingjian Ge, Nabil Hachem, Stavros Harizopoulos, John Lifter, Jennie Rogers, and Stan Zdonik. One Size Fits All? – Part 2: Benchmarking Results. 3rd Biennial Conference on Innovative Data Systems Research (CIDR), January 7-10, 2007, Asilomar, California, USA / Перевод Сергея Кузнецова, 2007 г.: [Электронный ресурс]. [http://citforum.ru/database/articles/one_size_fits_all_2/]. Проверено 27.06.2011.

4. Eric Lai. Size matters: Yahoo claims 2-petabyte database is world's biggest, busiest. 2008 г.: [Электронный ресурс]. [http://www.computerworld.com/s/article/9087918/Size_matters_ Yahoo_claims_2_petabyte_database_is_world_s_biggest_busiest]. Проверено 26.02.2012.
5. Григорьев Ю. А., Плутенко А. Д. Теоретические основы анализа процессов доступа к распределенным базам данных. – Новосибирск: Наука, 2002. – 222 с.
6. Григорьев Ю. А., Плужников В. Л. Оценка времени выполнения запросов и выбор архитектуры параллельной системы баз данных // Информатика и системы управления. – 2009. – № 3. – С. 3–12.
7. Григорьев Ю. А., Плужников В. Л. Модель обработки запросов в параллельной системе баз данных // Вестник МГТУ им. Н.Э. Баумана. Сер. Приборостроение. – 2010. – № 4. – С. 78–90.
8. Григорьев Ю. А., Плужников В. Л. Оценка времени соединения таблиц в параллельной системе баз данных // Информатика и системы управления. – 2011. – № 1. – С. 3–16.
9. Григорьев Ю. А., Плужников В. Л. Анализ времени обработки запросов к хранилищу данных в параллельной системе баз данных // Информатика и системы управления. – 2011. – № 2. – С. 94–106.
10. Григорьев Ю. А., Ермаков Е. Ю. Модель обработки запросов в параллельной колоночной системе баз данных // Информатика и системы управления. – 2012. – № 1. – С. 3–15.
11. Michael Stonebraker, Daniel J. Abadi, Adam Batkin, Xuedong Chen, Mitch Cherniack, Miguel Ferreira, Edmond Lau, Amerson Lin, Samuel R. Madden, Elizabeth J. O'Neil, Patrick E. O'Neil, Alexander Rasin, Nga Tran, and Stan B. Zdonik: C-Store: A Column-Oriented DBMS [Электронный ресурс]. [http://www.cs.yale.edu/homes/dna/pubs/displaypubs.cgi/]. Проверено 22.10.2011.
12. Daniel J. Abadi Query Execution in Column-Oriented Database Systems. [Электронный ресурс]. [http://www.cs.yale.edu/homes/dna/papers/abadiphd.pdf]. Проверено 25.12.2011.
13. Daniel J. Abadi, Daniel S. Myers, David J. DeWitt, and Samuel R. Madden. Materialization Strategies in a Column-Oriented DBMS In Proceedings of ICDE, 2007. [Электронный ресурс]. [http://db.lcs.mit.edu/projects/cstore/abadiicde2007.pdf]. Проверено 25.12.2011.
14. Daniel J. Abadi, Samuel R. Madden and Miguel C. Ferreira. Integrating Compression and Execution in Column-Oriented Database Systems In Proceedings of ICDE, 2006. [Электронный ресурс]. [http://db.lcs.mit.edu/projects/cstore/abadisigmod06.pdf]. Проверено 25.12.2011.
15. Соколинский Л. Б., Цымблер М. Л. Лекции по курсу "Параллельные системы баз данных": [Электронный ресурс]. [http://pdbs.susu.ru/CourseManual.html]. Проверено 04.12.2010.
16. Бахвалов Н. С. Численные методы (анализ, алгебра, обыкновенные дифференциальные уравнения). – М.: Наука, 1973. – 631 с.
17. AIDA64 Extreme Edition. [Электронный ресурс] [http://www.aida64.com/product/aida64-extreme-edition/overview] Проверено 08.04.2012.

Статья поступила в редакцию 28.05.2012